

## Homework #7

### Due by Friday 8/31 11:55pm

#### Submission instructions:

1. You should submit your homework in the NYU Classes system.
2. For this assignment, you should turn in 5 files:
  - Two '.cpp' files, one for each question 1-2.  
Name your files 'YourNetID\_hw7\_q1.cpp', and 'YourNetID\_hw7\_q2.cpp'.
  - One '.pdf' file with your answers for questions 3-7.  
Name your file 'YourNetID\_hw7\_q3to7.pdf'
3. Pay special attention to the style of your code. Indent your code correctly, choose meaningful names for your variables, define constants where needed, choose the most appropriate control flow statements, break down your solutions by defining functions, etc.

#### Question 1:

- a. Implement a function:

```
int printMonthCalender(int numOfDay, int startingDay)
```

This function is given two parameters:

- numOfDay - The number of days in the month
- startingDay - a number 1-7 that represents the day in the week of the first day in that month (1 for Monday, 2 for Tuesday, 3 for Wednesday, etc.).

The function should:

- Print a formatted monthly calendar of that month
- Return a number 1-7 that represents the day in the week of the **last day** in that month.

#### Formatting Notes:

- The output should include a header line with the days' names.
- Columns should be spaced by a Tab.

Example: when calling `printMonthCalender(31, 4)` it should return 6, and should print:

Mon	Tue	Wed	Thr	Fri	Sat	Sun
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

- b. A method for determining if a year is a leap year in the Gregorian calendar system is to check if it is divisible by 4 but not by 100, unless it is also divisible by 400. For example, 1896, 1904, and 2000 were leap years but 1900 was not. Write a function that takes in a year as input and return true if the year is a leap year, return false otherwise.

Note: background on leap year [https://en.wikipedia.org/wiki/Leap\\_year](https://en.wikipedia.org/wiki/Leap_year)

- c. Implement a function:

```
void printYearCalender(int year, int startingDay)
```

This function is given two parameters:

- `year` – an integer that represents a year (e.g. 2016)
- `startingDay` – a number 1-7 that represents the day in the week of 1/1 in that year (1 for Monday, 2 for Tuesday, 3 for Wednesday, etc.).

The function should use the functions from sections (a) and (b) in order to print a formatted yearly calendar of that year.

Formatting Note: As the header for each month you should print the months' name followed by the year (e.g. March 2016).

Example: Appendix A shows the expected output of the call `printYearCalender(2016, 5)`.

- d. Write program that interacts with the user and your function in (c).

### Question 2:

Consider the following definitions:

- a. A **proper divisors** of a positive integer ( $\geq 2$ ) is any of its divisors excluding the number itself.

For example, the proper divisors of 10 are: 1, 2 and 5.

- b. A **perfect number** is a positive integer ( $\geq 2$ ) that is equal to the sum of its proper divisors.

For example, 6 and 28 are perfect numbers, since:

$$6 = 1 + 2 + 3$$

$$28 = 1 + 2 + 4 + 7 + 14$$

Background of perfect numbers: [https://en.wikipedia.org/wiki/Perfect\\_number](https://en.wikipedia.org/wiki/Perfect_number)

- c. **Amicable numbers** are two different positive integer ( $\geq 2$ ), so related that the sum of the proper divisors of each is equal to the other number.

For example, 220 and 284 are amicable numbers, since:

$$284 = 1 + 2 + 4 + 5 + 10 + 11 + 20 + 22 + 44 + 55 + 110$$

$$220 = 1 + 2 + 4 + 71 + 142$$

*(proper divisors of 220)*

*(proper divisors of 284)*

Background of amicable numbers: [https://en.wikipedia.org/wiki/Amicable\\_numbers](https://en.wikipedia.org/wiki/Amicable_numbers)

- a. Write a function:

```
void analyzeDivisors(int num, int& outCountDivs, int& outSumDivs)
```

The function takes as an input a positive integer `num` ( $\geq 2$ ), and updates two output parameters with the number of `num`'s proper divisors and their sum.

For example, if this function is called with `num=12`, since 1, 2, 3, 4 and 6 are 12's proper divisors, the function would update the output parameters with the numbers 5 and 16.

Note: Pay attention to the running time of your function. An efficient implementation would run in  $\Theta(\sqrt{num})$ .

- b. Use the function you wrote in section (a), to implement the function:

```
bool isPerfect(int num)
```

This function is given positive integer `num` ( $\geq 2$ ), and determines if it is perfect number or not.

- c. Use the functions you implemented in sections (a) and (b), to write a program that reads from the user a positive integer `M` ( $\geq 2$ ), and prints:

- All the perfect numbers between 2 and `M`.
- All pairs of amicable numbers that are between 2 and `M` (both numbers must be in the range).

Note: Pay attention to the running time of your implementation. An efficient algorithm for this part would call `analyzeDivisors`  $\Theta(M)$  times all together.

**Question 3:**

What is the probability that a fair die never comes up an even number when it is rolled six times?

**Question 4:**

What is the conditional probability that a randomly generated bit string of length four contains at least two consecutive 0s, given that the first bit is a 1? (Assume the probabilities of a 0 and a 1 are the same).

**Question 5:**

Assume that the probability a child is a boy is 0.51 and that the sexes of children born into a family are independent. What is the probability that a family of five children has

- a) exactly three boys?
- b) at least one boy?
- c) at least one girl?
- d) all children of the same sex?
- e) the first child is a boy or that the last two children of the family are girls

**Question 6:**

Find the probability that a family with five children does not have a boy, if the sexes of children are independent and if

- a) a boy and a girl are equally likely.
- b) the probability of a boy is 0.51.
- c) the probability that the  $i^{\text{th}}$  child is a boy is  $0.51 - \frac{i}{100}$ .

**Question 7:**

Find each of the following probabilities when  $n$  independent Bernoulli trials are carried out with probability of success  $p$ .

- a) the probability of no failures
- b) the probability of at least one failure
- c) the probability of at most one failure
- d) the probability of at least two failures

## **Appendix A.**

The expected output of the call `printYearCalender(2016, 5)` is:

January 2016

Mon	Tue	Wed	Thr	Fri	Sat	Sun
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

February 2016

Mon	Tue	Wed	Thr	Fri	Sat	Sun
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29						

March 2016

Mon	Tue	Wed	Thr	Fri	Sat	Sun
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

April 2016

Mon	Tue	Wed	Thr	Fri	Sat	Sun
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

May 2016

Mon	Tue	Wed	Thr	Fri	Sat	Sun
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

June 2016

Mon	Tue	Wed	Thr	Fri	Sat	Sun
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

July 2016

Mon	Tue	Wed	Thr	Fri	Sat	Sun
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

August 2016

Mon	Tue	Wed	Thr	Fri	Sat	Sun
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

September 2016

Mon	Tue	Wed	Thr	Fri	Sat	Sun
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

October 2016

Mon	Tue	Wed	Thr	Fri	Sat	Sun
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

November 2016

Mon	Tue	Wed	Thr	Fri	Sat	Sun
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

December 2016

Mon	Tue	Wed	Thr	Fri	Sat	Sun
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	